TRW

(2)

Advanced Computing Systems: An Advanced Reasoning-Based Development Paradigm for Ada Trusted Systems and Its Application to MACH

ARPA Order No. 6414
Program Code No. 9T10
Contract No. MDA 972-89-C0029

Quarterly Status Report
Report #1
15 March 1989

TRW
FEDERAL SYSTEMS GROUP
Systems Division
2750 Prosperity Avenue
P.O. BOX 10400
Fairfax, Virginia 22031

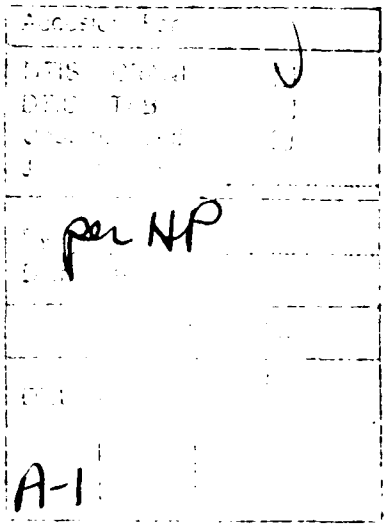
Quarterly Status Report
Report #1
15 March 1989

Reporting Period:
28 November 1988 - 28 February 1989

This Quarterly Technical Report provides information on our progress towards accomplishing our Phase I objectives. The body of this report consists of a set of attachments, primarily working notes, which provide snapshots of our technical work in progress.

Process Model Development

A major Phase I activity is the development of a process model for high performance trusted systems in Ada. The TRW/CLI/TIS team has been operating as a working group to develop the process model. We have been meeting as a working group every two weeks for 1-2 day "brainstorming" sessions and preparing material (briefings and/or internal working notes) in between sessions to further define issues we surface. The list below indicates the areas covered at the meetings we had during the reporting period..



- o 19 January 1989 Kickoff Meeting
  - Detailed schedule for Phase I Activities
  - Discussion of related activities
    - Ava work by CLI
    - Arcadia work by Arcadia Consortium
  - Review of documents for basis of process model:
    - Spiral Process Model
    - Ada Process Model (TRW use only)
- o 6 February 1989 Working Group Meeting
  - Overview of ASOS (Army Secure Operating System developed by TRW as A1 system in Ada)
  - Overview of Ava (A Verifiable Ada)
    - Ava Information
    - Ava and ASOS Ada (Attachment #3)
  - Process Model: Goals, Components and Drivers

- 17 February 1989 Working Group Meeting

  - ASOS Questions and Answers
  - Computer Security Lessons Learned
  - Risk Management for a trusted Application
  - Trust Engineering for Commercial Products (Attachment #4)
  - Process Model Brainstorming (Attachment #5)
  - Distinctions between commercial and Mission-critical trusted systems (Attachment #6)

Attachment #1 is a draft of the outline for our Process Model Report; Attachment #2 is a draft of an annotated outline for Volume I of the Report.

MACH Analysis

This initial analysis is described in Attachment #7, Ada/Security Study and MACH Analysis.

NOTE: ALL ATTACHMENTS TO THIS QUARTERLY REPORT ARE INTERNAL WORKING NOTES AND DRAFTS. THEY ARE FOR DARPA PROGRAM MANAGER USE ONLY AND ARE NOT FOR FURTHER DISTRIBUTION OR CITATION.

TRW

Attachment #1

PROCESS MODEL FOR HIGH PERFORMANCE TRUSTED SYSTEMS IN ADA

OUTLINE OF REPORT

Volume I

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Overview of Document

2. Objectives and Basis of the Process Model
   - 2.1 Objectives
   - 2.2 Basis/Foundation: Spiral Process Model

3. Motivations (Drivers) and Constraints of the Process Model
   - 3.1 Motivations (Drivers)
     - 3.1.1 Trust
     - 3.1.2 Ada
     - 3.1.3 Performance
   - 3.2 Constraints
     - 3.2.1 Political/Sociological Environment
     - 3.2.2 Cost
     - 3.2.3 Available Technology/Knowledge

4. Overview of Primary Risk Issues
   - 4.1 Trust-Related Risks
     - 4.1.1 Technological Immaturity
     - 4.1.2 Policy Risks
     - 4.1.3 Assurance Risks
     - 4.1.4 Development Risks
     - 4.1.5 Evaluation/Accreditation Risks
   - 4.2 Ada-Related Risks
     - 4.2.1 Technological Immaturity
     - 4.2.2 Ada-Inexperienced Staff
     - 4.2.3 Inadequate Resources
   - 4.3 Performance-Related Risks
     - 4.3.1 Mission Inadequacy
     - 4.3.2 Evolvability
     - 4.3.3 Interactions with Trust

PROCESS MODEL FOR HIGH PERFORMANCE TRUSTED SYSTEMS IN ADA

OUTLINE OF REPORT

Volume II

1. Distinctions Between Trusted Commercial Product and Mission-Critical Systems Development
2. Application of Process Model to Trusted Commercial Product
3. Application of Process Model to Trusted Mission Critical System
4. Application of Process Model to Trusted Mach in Ada
5. Tailoring of Process Model to Dod-STD-2167A

Appendices

A: Trust Requirements for the Software Development Environment

B: Ada/Security Study and Mach Analysis

C: *Initial Specification of Trust Analysis Tool (for Arcadia Environment?)*

TRW

Attachment # 2

DRAFT
13 March 1989

PROCESS MODEL FOR HIGH PERFORMANCE TRUSTED SYSTEMS IN ADA

ANNOTATED OUTLINE OF REPORT

Volume I

1. Introduction

   1.1 Purpose
   - Increase productivity and quality of high performance trusted systems
   - Reduce risks inherent in trusted system development
   - Incorporate lessons learned
   - Exploit features of Ada (language and software engineering aspects of it)
   - Effectively integrate security, trust and performance engineering with a modern system development paradigm for Ada

   1.2 Scope
   - Definition of a process model (including components)
   - Activities addressed by the process model
   - Types of systems to which it is intended to be applied
   - Timeframe we're addressing -- long term vision (mid-1990's and beyond) but guidance on nearer term application (examples in Volume II)
   - Guidance based on extensive experience and lessons learned but not a "cookbook"
   - Relationship to standards and policies (TCSEC, 2167A, company and government policies)
   - Applicability for non-Ada projects

   1.3 Overview of Document
   - Brief description of what is in each of the rest of the chapters in Volume I
   - Brief description of Volume II
   - Brief description of the Appendices

2. Objectives and Basis of the Process Model

   2.1 Objectives
   - Expand purpose described very briefly in Section 1.1 and give an overview of how elements of the process model satisfy its primary goals, e.g.,
     - Reduce risks
     - Increase productivity
     - Improve quality
     - Achieve high performance and trust

2.2 Basis/Foundation: Spiral Process Model
- Overview of the spiral process model and its key aspects (include diagram)
- Foundation into which we've integrated security, trust and performance engineering, Ada and formal methods, etc. in the context of sound software engineering practices and automation and support
- Stress the following aspects:
    - Improving process and product
    - Risk management throughout
    - Iteration at different levels and risk reduction activities can be carried out in parallel

3. Motivations (Drivers) and Constraints of the Process Model

3.1 Motivations (Drivers)

Primary motivations for developing the process model fall into three categories:
- Trust-based motivations
- Ada-based motivations
- Performance-based motivations

3.1.1 Trust
- Definition and aspects of trust
- Continuum of trust levels
- Trust dimensions diagrams and discussions — from brainstorming
- Metrics on trust — notes of 2.17.89
- Tradeoffs against trust
- Reasoning needed for trust
- Configuration management and control for trust

3.1.2 Ada
- Applicability of Ada as a notation for multiple activities of the lifecycle (uniformity in specification/design and implementation)
- Potential for Ada reuse
- Software engineering aspects of Ada
- Opportunity to use tools, e.g., to support reasoning and configuration management and control for trust
- Ada as a specification language
- Verifiability of Ada
- Customers' commitment to Ada

3.1.3 Performance
- Aspects of performance
    - Speed
    - Effectiveness
    - System performance
    - Mission-criticality performance
    - Resource utilization
    - Real-time effects
- Reasoning for performance

- Design tension and interaction with trust

3.2 Constraints

Primary constraints impacting the process model fall into three categories:
- Political/sociological environment (today and of the mid 1990's)
- Cost
- Available technology/knowledge

3.2.1 Political/Sociological Environment
- Overview of distinctions between commercial and mission-critical trusted systems
- Notes from 2/17/89 brainstorming session (collapsed into less categories and less information)

3.2.2 Cost
- Cost-effective
- Financial rules
- Different cost distribution
- Appropriate use of assurance technology consistent with cost
- Schedule impacts

3.2.3 Available Technology/Knowledge
- Recognition of where we are and projection of where technology and knowledge will be in 1990's
- Available products/components
- Support tools
- Maturity level
- Trustworthiness of the development environment

4. Overview of Primary Risk Issues

Discussion of the key risk aspects of each of the three drivers impacting the process model and the issues associated with controlling them.

4.1 Trust-Related Risks

4.1.1 Technological Immaturity
- Conceptual risks of ill-understood issues and unsolved problems
- Development risks: design and implementation techniques, principles, metrics not well established and few good examples
- Lack of leverage to build on top of existing trusted hardware and software

4.1.2 Policy Risks

- Ability to model broader notion of trust than confidentiality
- Policy model satisfy policy objectives? and is it useful and usable?
- Features to implement policy
  - Adequate to support application developers and system users
  - Incompatible with existing commercial components or customer developed software base
  - Difficulties in demonstrating correspondence between policy and implementation

4.1.3 Assurance Risks

- Size, simplicity and isolation of the trusted computing base
- Risks of analyses and verification
  - Value of verifying formal model vs. implementation verification
  - Level of testing appropriate
  - Feasibility and meaningfulness of code correspondence
  - Lack of tools to effectively support analyses and verification of Ada throughout lifecycle
  - Completeness and correctness of covert channel analysis and penetration testing and analysis

4.1.4 Development Risks

- Prioritization to be carried out throughout lifecycle of cost, schedule, trust, performance, mission and functionality, etc. -- needs to be consistent
- Skill specialization between security/trust engineers and system developers
- Redundancy and contradictions in both documentation and in system as it evolves and develops
  - System requirements vs. trust policy and rationale
  - Evolving Ada design, specifications and code and how it relates to formal and descriptive top level specifications
  - Testing plans and results vs. security testing
  - Configuration management to control and manage the system and the documentation
- Confusion over extensions/constraints of methodology and development standards for trusted computing base
  - Design rules, coding standards, naming conventions
  - Configuration management and testing requirements
  - Specialized tools may be needed
- Redevelopment (maintenance) activities may invalidate assurances or trigger recertification or reevaluation

4.1.5 Evaluation/Accreditation Risks

- Definition of evaluation and accreditation and who performs each
- "Rules of the game" not well-defined or stable
- Pushing the state-of-the-art requires new interpretations of TCSEC and other evaluation criteria for broader definitions of trust
- Incremental evaluation not done before
- Conflicting demands of accreditor/evaluator, customer and end-user

4.2 Ada-Related Risks

4.2.1 Technological Immaturity

- Immature compilers may result in programs not compiling or incorrect compilation
- Immature support system
  - Run-time support may be incompatible with host environment
  - Tools may be missing or full of errors
  - Tools may be incompatible or inefficient
- Inadequate support for use of Ada throughout the lifecycle
  - Immature or no tools to support Ada as a design language or Ada as a specification language
  - Inadequate support for reasoning for trust and performance for Ada
  - Configuration management and control for entire lifecycle is lacking

4.2.2 Ada-Inexperienced Staff

- Advanced Ada features are seductive and easy to misuse, e.g., tasking, generics and limited private types
- Too easy to overassess Ada advantages, e.g., portability of code, compiler catching more errors
- Inexperienced Ada managers need to focus more on value of front end investment, different staffing needs, hardware selection/software impact risks
- Little experience in industry in use of Ada for trusted systems

4.2.3 Inadequate Resources

- Ada compilers do more and need additional computing power
- More mass memory required
- Training can be significant resource - consumer
- Immaturity of tool suppliers can cause schedule problems
- Access to "Ada gurus" is critical scarce resource
- Mismatch between pre-Ada budget and schedule and reality of Ada developerments

4.3 Performance-Related Risks

4.3.1 Mission Inadequacy

- Inadequate attention to performance issues throughout development lifecycles may cause system unable to perform mission and not meet performance requirements
- Inability to identify mission risks where failures would result in serious mission compromise
- Inadequate integration of techniques to address performance/mission risks result in risks not being mitigated

4.3.2 Evolvability

- Inadequate attention to system evolvability may result in system not achieving desired high performance

4.3.3 Interactions with Trust

- Keeping both trust and high performance as priorities is difficult in system engineering, design engineering and development
  - Access mediation vs. throughput and responsiveness
  - Always trading off one against the other throughout the lifecycle
  - Engineering and development techniques to achieve performance often impact trust and vice versa
- Engineering for both is key challenge
- Configuration management for performance and trust so we have adequate visibility and metrics is important risk area to be addressed

5. Elements of the Process Model

- Process model is an integration of strategic elements from both research and practice
- Process model has long-term vision but also nearer-term applicability

5.1 Risk Management

- Highlight risks throughout lifecycle activities
- Software engineering and management techniques for managing and controlling risks
- Develop risk management plan
  - Do it early
  - Maintain it throughout lifecycle; it evolves
  - Address primary risks first
  - Include decisive risk mitigation actions and link to transition criteria

5.1.1 Formal Risk Management Techniques

- Identify (generic) primary risks in building high performance trusted systems (based on Section 4) early in the process, evaluate the risks, decide on action to reduce risk (e.g., modeling, prototyping, analysis, etc.)
- Risk identification/evaluation performed at beginning of each major activity of development
- Specific actions mitigate specific risk areas

5.1.2 Modeling and Specification

- Identify key aspects of trust and performance to be modeled, informally and then formally, to a level that's appropriate for the system development and the criticality of the risk
- Modeling and specification activities help identify unknown or fuzzy requirements, flesh out critical performance issues and mitigate policy risks as well as early development risks
- If performed concurrently with prototyping, helps reduce risk of redundancy and contradictions with the evolving Ada design and development

5.1.3 Prototyping and Demonstrations

- Identify and mitigate key risks associated with fuzzy requirements (especially with respect to performance and trust), system architecture definitions, critical mechanisms for trust and performance, user interface (as appropriate for system being built)

- Prototypes reduce overall project risks while supporting the development process; reduce risk that a particular technical approach will not satisfy performance requirements
- Use of Ada for prototyping and demonstrations helps validate interfaces early, a high risk issue
- Prototyping helps analyze performance and evaluate evolvability of the system
- Evaluate prototype "product" and incorporate lessons learned into the design (when Ada used for prototyping, gives traceability and early compilable results for later development)

5.1.4 Incremental Development

- Process model utilizes incremental software design and development approach which provides early availability of operational software and better visibility into the software development process
- Early incremental builds defined to address areas that are higher risk, prerequisites to later incremental builds or required for early integration and test activities
- Approach gives increased flexibility and facilitates early risk reduction since (high risk) less mature system capabilities may be implemented early for advance warning of potential or real project risks

5.2 Ada in the Process Model

Process model does not limit Ada's role to the implementation activity; Ada is used across multiple activities and reuse of Ada components encouraged as well as a consistent use of metrics across the lifecycle.

5.2.1 Homogeneous Representation

- Ada used as a design language and potentially as a (formal) specification language in addition to being used as the implementation language
- Ada's abstraction and language extensibility features enable description of software design; at each level of design, complete Ada program can be created using Ada specifications to formally define interfaces among objects and operations
- Can include some design information as comments in the Ada code throughout lifecycle since design expressed as compilable Ada design language and then transformed into code
- Advantages
  - Compiler can be used early in design process for automated syntactic and semantic checking of logical inconsistencies
  - Consistent set of tools can be used from preliminary design on
  - Maintaining design and configuration management of design simplified
  - Validating design supported by compiler which results in lower risk and earlier resolution of interface problems
  - Enables better metrics to be used
- Ada as specification language (needs further elaboration)

5.2.2 Metrics

- Benefit of using Ada throughout is the consistency of metrics to determine design progress, product growth and requirements change impacts
- Metrics can be compared against planning estimates to evaluate progress and analyze trends -- get earlier visibility and better insight into potential later problems
- Enable useful metrics to be generated

5.2.3 Reuse

- Process model encourages consideration of available components that can be reused at all levels of design
- Expect that reusable Ada components are more mature, known and validated quantities and less risky (??) to use
- Impact of reusable software assessed several times during lifecycle starting with software architecture definition
- Taking advantage of reuse (assuming there is lots of good stuff in reuse libraries to reuse) can increase software quality because software becomes mature through reuse and further analysis
- Reuse in the face of trust requirements and performance requirements is a significant challenge

5.3 Engineering for Trust and Performance

- Trust and performance must be kept as equal high priority requirements because they are often in opposition to one another

5.3.1 Architecture Analysis

- Throughout lifecycle, assessment of system architecture carried out to determine support for trust and performance requirements
- Initial trust assessment determines or evaluates hardware base as well as software architecture for its structure and its support of access mediation
- Structure and characteristics of potential trusted computing base determined or evaluated for adequacy for trust and performance requirements
- Critical architecture mechanisms and skeletons are prototyped to enable analysis for further development and engineering -- validates interfaces, evaluates growth and flexibility
- Modeling and specification of critical mechanisms can also support trust and performance engineering

5.3.2 Performance Modeling

- Performance engineering techniques (more details later) applied throughout the system lifecycle
- Discrete analysis and modeling techniques support achieving high performance requirements
- Throughout lifecycle, assessment of desired performance characteristics and allocation of performance budgets to components
- Initial performance assessment emphasis on determining what portion of the resource use is attributable to trust mechanisms
- Information flow analysis to determine performance bottlenecks

- Discrete analysis and performance modeling techniques to support bottleneck reduction and "what if" alternative architecture analysis
- Prototyping of alternatives to high impact bottlenecks or proposed trust mechanisms

5.4 Other Software Engineering Techniques

- Two categories of software engineering techniques are essential elements of the process model: various analysis and assurance techniques as well as configuration management and control techniques

5.4.1 Analysis/Assurance Techniques

- Process model effectively integrates spectrum of analysis and assurance techniques, both informal and formal, to support design, development and assessment of high performance trusted systems
- Techniques include initial criticality assessment, trust and performance assessment, informal and formal specifications at appropriate levels of abstraction, various levels of testing, covert channel analyses, security and penetration testing, informal and formal modeling, performance modeling **[section needs to be expanded and a diagram would be useful here]**

5.4.2 Configuration Management and Control

- Since spiral process model allows so much flexibility and various design and development activities are proceeding in parallel (e.g., prototyping and modeling) but to differing levels of detail and completion, it is essential to have strong configuration management and control from the earliest time possible, e.g., as soon as there are compilable Ada specifications or earlier
- Effective (automated) configuration management and control throughout the lifecycle highly advantageous -- early visibility and control, early risk reduction because of problems identified earlier

5.5 Accreditation and Certification

- Since the objective of the process model is development of high performance trusted systems, accreditation and certification (of some sort) performed external to the system development process yet design, documentation and assurance components of the process model are needed to support the accreditation and certification process
- High level description of the accreditation and certification process useful here as well as the components of the process model needed to support this process
  - Architectural design
  - Documentation
  - Assurance

6. Process Model in the Lifecycle **[More work to be done in annotating section]**

- More effort required during analysis and design activity than generally required by traditional system development
- Adjustments in project planning and scheduling necessary

- Small expert team used effectively during requirements and design activities to work critical risk issues
- Several aspects of process model facilitate small team approach but a great deal of communication on all fronts necessary
- Process model divides lifecycle into nine activities
    - Activities, not phases, to emphasize iterative nature of the activities within process model, rather than sequential approach taken earlier
    - Various activities may be done in parallel but some are best done sequentially (we will describe dependencies and inter-relationships)
- Activities addressed:

    - Concept Definition
    - Trust/Performance Assessment
    - Requirements Analysis
    - Analysis
    - Modeling and Specification
    - Prototyping and Demonstrations
    - Preliminary Design
    - Detailed Design
    - Implementation and Integration
- Risk reduction and configuration management to be done throughout lifecycle
- Elements of the process model introduced in Section 5; this section provides a summary of the key elements of the process model in the context of the activities of the lifecycle
- For each of the lifecycle activities, we will define (generic) risks addressed, risk mitigation strategies, approaches and tasks, typical documents or prototypes produced, results and criteria to transition to other lifecycle activities

6.1 Concept Definition

- Activities consist of identifying system concept for the project focusing on critical risk issues and determination of system feasibility

6.1.1 Critical Risks

- Identification of primary trust, performance and other risks in the system concept definition
- Perform studies and analyses and prepare reports, e.g.
    - Threat analysis
    - Operational analysis
    - Initial trust assessment
    - Definition of environmental characteristics
    - Performance/trust issues

6.1.2 System Feasibility

- Perform analysis to determine if early identified system concept requirements can be met
    - Identify and resolve trust, performance, budget and schedule objectives
    - Determine analysis/assurance technology needed
- Prepare:
    - Initial Risk Management Plan

- System concept document
- Strawman budget and schedule

6.2 Trust/Performance Assessment

6.2.1 Criticality Assessment

- Further analyzes critical risk issues and determines trust level for system to try to achieve
- Determines significant performance impacts to address in risk management plan
- Results in initial set of high level critical system requirements

6.2.2 Architecture Assessment

- Determine whether the architecture proposed (or to be enhanced) can achieve the desired trust level and achieve the desired performance requirements
- Analyze hardware base and the software architecture for structure and its support of access mediation
- Initial high level prototyping may be performed to determine COTS use; reuse assessment performed
- Hardware performance modeling performed

6.2.3 Documentation

- If a system enhancement, design documentation analyzed to determine *support for the architecture analysis and assurance requirements* for the trust level to be achieved
- Security/trust policy document developed

6.3 Control and Management

- These activities are carried out throughout the lifecycle to manage and control the process model efforts so that there is review and concensus before transitioning to the next set of activities; also used to configuration manage the evolving portions of the system in various stages of modeling, prototyping and analyzing

6.3.1 Configuration Management and Control

- System being developed evolving and various activities performed in parallel but to different degrees of completion
- Version history kept, prototypes tracked, compilable Ada "components" configuration managed so that visibility in progress available and closure on process and products achieved

6.3.2 Reviews and Walkthroughs

- Reviews and walkthroughs are used throughout lifecycle to achieve consensus or perform iteration on an activity if concensus not reached
- This section will describe the various reviews and walkthroughs to be carried out, the decisions made and the documents and system components to be produced which are reviewed **[diagrams useful here]**

6.4 Requirements Analysis

- Establishes set of requirements based on the initial concept definition and trust/ performance assessment; governed by risk management plan
- Primary activities include: **(needs elaboration)**
  - System analysis
  - Performance analysis
  - Software reuse assessment
  - Security/trust model development
  - Trade-off analyses
  - Growth and flexibility scenarios
  - Prototyping or modeling for early COTS and architecture issue resolution
  - Formulation of early risk/vulnerability analysis

6.5 Analysis

- Process model integrates spectrum of analysis and assurance techniques aimed at supporting the reasoning that is needed for trust and performance as well as supporting the accreditation and certification activities to be performed
- Techniques Include: **[Needs elaboration]**
  - Analyses and trade-off studies performed (described in Sections 6.1, 6.2 and 6.4)
  - Informal and formal trust and performance model
  - Informal and formal verification of design and implementation
  - Risk/vulnerability assessments
  - Covert channel analyses
  - Various levels of testing throughout
  - Penetration testing
  - Performance and security testing

6.5.1 Reasoning for Trust

- This section will describe the various analysis and assurance techniques that support trust reasoning
- For each technique, define what it is, what risk area it addresses, when it is performed, other analyses it relates to, role it plays, transition criteria for next activity

6.5.2 Reasoning for Performance

- This section will describe the same characteristics for the analysis and assurance techniques that support the performance reasoning (e.g., performance modeling and simulation, critical mechanisms prototyping, algorithm modeling, etc.)

6.6 Modeling and Specification

- Process model incorporates both informal and formal modeling and specification activities carried out during the lifecycle to address specific risk areas and provide assurances in support of building high performance trusted systems
- Techniques include: **[Needs elaboration]**
  - Security/trust model
  - Performance model

- Descriptive top level specification
- Formal top level specification

- Process model encourages use of Ada as a specification language; this section will discuss how Ada fits into the specification activities

6.6.1 Informal

- This section will describe the informal modeling and specification, what risk issue it addresses, when it is performed, role it plays, transition criteria for next activity

6.6.2 Formal

- Same description for formal modeling and specification techniques

6.7 Prototyping and Demonstration

- Critical risk management strategy is construction of prototypes of both the system architecture skeleton and the critical mechanisms of the system (early use of Ada desirable)
- Provide initial strawman for evaluating the system solution and tor surfacing the architecture and critical mechanism limitations early in the lifecycle

6.7.1 Architecture Skeleton

- This early prototyping activity helps evaluate critical risk drivers and assesses whether reuse of existing components possible as well as support for trust and performance requirements
- Prototypes begin from top-level interfaces and components to provide a platform for incremental integration of the subsystems as they evolve (user interface prototyped if high risk issue)
- Architecture skeleton represents the top-level software design into which lower level modules can be added incrementally
- Provides early evaluation of candidate approaches, demonstrable initial testbed environment for review and feedback, early tangible evidence of performance and use of actual interfaces to validate them early

6.7.2 Critical Mechanisms

- Provides a means for evaluating trust, functionality and performance characteristics of certain mechanisms determined by earlier analysis to be critical
- These may be algorithms or security mechanisms that can greatly impact performance if not designed properly

6.8 Design and Implementation Approach **[Needs elaboration and diagrams]**

6.8.1 Incremental Development

- Process Model approach to software development emphasizes development of incremental capabilities with incremental generation and review of the design and documentation products
- Provides early availability of operational software and additional visibility into development process

- Early increments address areas that are high risk **(examples here?)**, precursors to later development, or required for early software development and integration; later increments provide additional software functionality until full software structure complete
- Description of walkthroughs and review?

6.8.2 Design Validation **[Needs elaboration and diagrams]**

- Use of Ada for design permits automated tools (e.g., compiler) to be used early to validate syntax and some semantics

6.9 Accreditation and Certification

- This section will describe the general accreditation and certification activities of the process model [TCSEC activities only one aspect of it]

Glossary

This section will define key terminology used in the report.

Attachment #3

Ava and ASOS Ada

John McHugh

6 February 1989

Overview

Ava is a subset of Ada for which a formal definition is being developed. Several formalisms are being used.

- Denotational Semantics
- Boyer Moore Logic
- Lisp

The ASOS operating system was developed using a subset of Ada dictated by a variety of security related factors. A significant factor is the ASOS use of Gypsy specifications for its FTLS and the need for showing correspondence between the Gypsy FTLS and the Ada code.

We hope to compare these subsets and determine whether Ava is *or can be made to be* suitable for trusted operating system development.

Sources

The following documents form the basis for the comparison:

1. *The Ava Reference Manual: Derived from ANSI/MIL-STD-1815A-1983* CLI Draft Technical Report Version A2. Jan 1989
2. *The nanoAVA Definition* CLI Technical Report 21, Revised Draft, Oct. 1988
3. *Initial Development Specification Rationale for ASOS* TRW ASOS Contract CDRL G014 (Draft) 8 April 1987
4. *MLS OS Final Development Specification Rationale for ASOS* TRW ASOS Contract CDRL G015 (Draft) 12 May 1988
5. *Reference Manual for the Ada Programming Language - ANSI/MIL-STD-1815A-1983* Feb. 1983 U.S. Dept. of Defense

Organization

The presentation follows the order of the Ada Language Reference Manual (for better or for worse). Each section (or subsection as necessary) will be summarized with a slide for Ava followed by a slide for the ASOS subset and a slide for discussion if the differences seem to warrant.

References to the Ada Language Manual will be of the form [LRM i.j..-n] where i,j, etc. are chapter, section, etc. and n is a marginally numbered paragraph.

Ava - Chapter 1

The purpose of the Ava manual is to facilitate proofs of the correctness of programs written in the Ava subset.

[LRM 1.3] "Design Goals and Sources" has been removed.

[LRM 1.4-12] "Goto" statements have been removed.

[LRM 1.4-17] Statements applicable to tasking have been removed.

[LRM 1.4-20 and 25] Access types have been removed.

[LRM 1.4-22] Numeric types FLOAT and DURATION have been removed.

Ava - Chapter 1 cotd.

[LRM 1.4-24 and 26 and 27] Discriminated records have been removed.

[LRM 1.4-29] Representation clauses and Machine Code insertions have been removed.

[LRM 1.4-31] Generics have been removed.

[LRM 1.6-6.10] Erroneous execution, Incorrect order dependencies, and optional PROGRAM-ERROR compilation have been removed.

ASOS - Chapter 1

Neither of the ASOS sources explicitly covers this Chapter.

Discussion - Chapter 1

The removals of section 1.4 will be discussed later. The removal of erroneous execution and incorrect order dependencies moves a burden from the programmer to the programming system.

This removal may affect the compilability of Ava by Ada compilers due to strengthened requirements on the order of evaluation of Ava programs.

Ava - Chapter 2

[LRM 2-1] References to Pragmas have been removed.

[LRM 2.4] All references to real literals have been removed.

[LRM 2.8] Pragmas - removed.

ASOS - Chapter 2

No changes. It is not clear if ASOS actually makes use of Pragmas or of real types.

Discussion - Chapter 2

Note that AVA does not "unreserve" any Ada reserved words. The formal definition will make programs containing reserved words whose meaning has been removed from the Ava definition illegal.

Pragmas could be used to restrict compiler behavior when compiling Ava programs. In this case, they would have no meaning in Ava but could be used to ensure that a compiler having an Ava mode enforced certain rules that ensured conformance to the Ava definition.

Ava - Chapter 3

[LRM 3.1] References to declarations germane only to discriminants, tasks, and generics have been removed.

[LRM 3.2] References to Objects germane to reals, tasks, discriminants, and generics removed. References to Objects defined in terms of access types and slices removed.

[LRM 3.2-9] Objects must be initialized.

[LRM 3.2.1] Object Declarations - Discussions of implicit initialization removed.

[LRM 3.2.2] Number Declarations - Discussion of reals removed.

[LRM 3.3] Types and Subtypes - Reals, Tasks, Discriminant and Derived type references removed.

[LRM 3.4] Derived types - Omitted

Ava - Chapter 3 cotd.

[LRM 3.5.4-7] Predefined integer types except INTEGER removed.

[LRM 3.5.5] Attributes WIDTH, and IMAGE removed. (-4 and -10,11)

[LRM 3.5.5-14] removed (Object attributes A'SIZE and A'ADDRESS)

[LRM 3.5.6] Real Types - Removed

[LRM 3.5.7] Floating Point Types - Removed

[LRM 3.5.8] Operations of Floating Point Types - Removed

[LRM 3.5.9] Fixed Point Types - Removed

[LRM 3.5.10] Operations of Fixed Point Types - Removed

Ava - Chapter 3 cotd.

[LRM 3.6.2] Operations on Arrays - Attributes T'SIZE, A'SIZE and A'ADDRESS omitted. Relational and logical operations omitted.

[LRM 3.7] Variants and discriminants omitted

[LRM 3.7.1] Discriminants - Removed

[LRM 3.7.2] Discriminant Constraints - Removed

[LRM 3.7.3] Variant Parts - Removed

[LRM 3.7.4] Operations of Record Types - The Attributes A'CONSTRAINED, T'SIZE, A'SIZE and A'ADDRESS are omitted.

[LRM 3.8] Access Types - Removed

[LRM 3.9] Declarative Parts - References to tasks and generics omitted.

ASOS - Chapter 3

ASOS permits access types, but prohibits the NEW operation which somewhat constrains them. Need to determine to what extent they are actually used in the kernel.

ASOS permits Variant records and discriminants.

ASOS permits reals but attempts to minimize their use. Need to determine actual usage.

Discussion - Chapter 3

There seems to be no real technical objection to the inclusion of Variant records in Ava. They are omitted for now to reduce complexity. They will probably be required to accurately implement MACH message structures.

The question of access types is more difficult. The Ava group at CLI is strongly opposed to admitting them. At the same time, it seems that they present no deep technical problems. MACH depends heavily on address variables for process management and communications and it is difficult to conceive of an efficient implementation without them.

Reals are best left alone for now. Fixed point numbers are worth a second look, but appear to have no impact on MACH.

Ava - Chapter 4

[LRM 4.1] References to tasks, generics, reals, etc. omitted.

[LRM 4.1.2] Slices - Removed.

[LRM 4.2] References to real literals omitted.

[LRM 4.3-4] Named and positional associations cannot be mixed.

[LRM 4.5] Operations on reals are omitted. Logical (but not comparison) operations on aggregates are omitted.

[LRM 4.6] Type conversions involving reals, discriminants, and access types are omitted.

[LRM 4.8] Allocators - Removed

[LRM 4.9] Statics - References to reals and discriminants removed.

Ava - Chapter 4 cotd.

[LRM 4.10] Universal Expressions - References to reals removed.

ASOS - Chapter 4

No particular restrictions

Discussion - Chapter 4

None of the Ava restrictions appear to cause any particular difficulties.

Ava - Chapter 5

References to real, task, and generic related items are removed throughout the chapter.

[LRM 5-2] References to task related and code statements removed.

[LRM 5.1] References to LABEL removed. GOTO statement removed.

[LRM 5.2] Case in which type of RHS used to determine type of LHS (AI-00120) does not occur in Ava.

[LRM 5.3] Array slice assignment omitted.

[LRM 5.9] GOTO statement - Removed.

ASOS - Chapter 5

No specific restrictions. (Are gotos actually used?)

Discussion - Chapter 5

No problems anticipated.

Ava - Chapter 6

[LRM 6-1] Reference to tasks and generics omitted.

[LRM 6.1] Reference to default expressions for formal parameters omitted.

[LRM 6.2] Formal parameter mode OUT is omitted. References to constraints and discriminants are omitted. Much of the discussion of potentially erroneous programs is omitted.

[LRM 6.3] Programs written in a different language are omitted.

[LRM 6.3.1-2,4] Literal substitutions omitted.

[LRM 6.3.2] Inline expansion - Removed.

Ava - Chapter 6 cotd.

[LRM 6.4] Named parameter associations removed. Aliasing explicitly prohibited.

[LRM 6.4.1] Mode OUT, access types, and discriminants omitted.

[LRM 6.4.2] Default Parameters - Removed.

[LRM 6.6] Overloading of operator symbols by user subprograms omitted.

[LRM 6.7] Overloading of Operators - Removed.

ASOS - Chapter 6

ASOS prohibits aliasing and functions that reference global variables or that have side effects. Procedures are allowed to reference global state variables.

Discussion - Chapter 6

The prohibition of OUT mode parameters is tied in with ensuring that all variables always have a well defined value. The prohibition against aliasing eliminates a class of erroneous programs.

The omission of named parameter associations and default parameter values appears inconsequential, but unnecessary except to simplify the definition.

Ava - Chapter 7

Changes to this chapter are primarily devoted to omitting references to Ada features already removed from Ava such as tasks, generics, discriminants, reals, operator symbol overloading, programs in other languages, etc.

[LRM 7.6] Text Handling Example - Removed.

ASOS - Chapter 7

No specific restrictions.

Discussion - Chapter 7

None

Ava - Chapter 8

As seen earlier, it is possible to interpret this chapter so as to preclude legal Ada programs. The Ava version is almost identical to the Ada version except for reference to a number of AIs.

On the other hand, formalizing this chapter is a major headache. One problem that arises is the possibility that other Ava omissions render unambiguous overload resolution problems that would be ambiguous in full Ada.

It is to be expected that this chapter will change as the formalization progresses.

ASOS - Chapter 8

ASOS prohibits renaming declarations.

Discussion - Chapter 8

None

Ava - Chapter 9

[LRM 9] Tasks - Removed

ASOS - Chapter 9

ASOS prohibits Ada tasking in the kernel. It permits it in non-kernel trusted software.

Discussion - Chapter 9

Ada tasking is complex and not well understood. Most implementations provide their own task monitors so that OS tasking facilities are not used. It does not appear to be easy to implement Ada tasking in Ada thus making it difficult to verify the support even if Ava verification were available.

Clearly, a better model will be necessary for MACH.

Ava - Chapter 10

Changes to this chapter are primarily devoted to omitting references to previously omitted features of the language such as pragmas and subunits.

Problems concerning the recompilation of Library units and orders of elaboration interact with visibility rules and complicate the formalization.

ASOS - Chapter 10

No Impact

Discussion - Chapter 10

Probably cries out for some simplifying assumptions. Look at GVE incremental development model MK-II.

Ava - Chapter 11

References to tasks, subunits, reals, generics, etc. have been omitted throughout the chapter.

[LRM 11.1-6] NUMERIC_ERROR is no longer implicitly raised. CONSTRAINT_ERROR is raised instead {AI-00387}

[LRM 11.3-2] A RAISE statement must explicitly name the exception being raised. Implicit propagation is not possible.

[LRM 11.5] Exceptions During Tasking - Removed.

[LRM 11.6] Exceptions and Optimization - Removed.

[LRM 11.7] Suppressing Checks - Removed.

ASOS - Chapter 11

ASOS avoids exceptions at the TCB boundary to simplify covert channel analysis. It requires handlers and explicit propagation elsewhere.

Discussion - Chapter 11

The Ava restrictions do not seem to preclude careful and disciplined use of exceptions. The interaction with parameter passing mechanisms seem to preclude meaningful interpretation of the results of abandoned computations, but the formalization will be the final arbiter of that.

Ava - Chapter 12

[LRM 12] Generics - Removed.

ASOS - Chapter 12

Generics not permitted.

Discussion - Chapter 12

Dave Musser has been doing a lot of work in this area. Validated compilers produce wildly differing results on many examples. Probably best to avoid it for now, but it looks so nice in many ways.

This is probably the point at which the complexity of Ada collapses on itself.

Ava - Chapter 13

[LRM 13.1] Representation Clauses - Removed

[LRM 13.2] Length Clauses - Removed

[LRM 13.3] Enumeration Representation Clauses - Removed

[LRM 13.4] Record Representation Clauses - Removed

[LRM 13.5] Address Clauses - Removed

[LRM 13.5.1] Interrupts - Removed

[LRM 13.6] Change of Representation - Removed

Ava - Chapter 13 cotd.

[LRM 13.7] Package SYSTEM is reduced to the minimum and maximum integers and the system name.

[LRM 13.7.1] Named Numbers are MIN_INT and MAX_INT

[LRM 13.7.2] Representation Attributes - Removed

[LRM 13.7.3] Representation Attributes of Real Types - Removed

[LRM 13.8] Machine Code Insertions - Removed

[LRM 13.9] Interface to Other Languages - Removed

[LRM 13.10] Unchecked Programming - Removed

ASOS - Chapter 13

ASOS allows but restricts system dependent features. This is necessary in an OS that is to run on bare hardware.

Discussion - Chapter 13

Some mechanism will have to be developed to either interface with machine code or to handle interrupts and representations or both.

Thought should be given to integrating the Ava model with machine models such as those used for KIT.

Ava - Chapter 14

To be done in the future

ASOS - Chapter 14

ASOS minimizes use of Ada I/O.

Discussion - Chapter 14

Ava will eventually have a definition for at least some portion of the Ada I/O package. The same arguments that apply to ASOS would apply to MACH. We need to develop the primitive I/O within the system, not outside it.

Conclusion

The Ava definition is supposed to be finished by the end of the year.

Ava will probably have most of the features needed for MACH server or kernel implementation. We will need to push in some areas, notably Variant structures and access types.

Marty Branstad
2/17/89
attachment #4

TRUST ENGINEERING

- New Operating System
- Modified Operating System
- Application System

Focus on B2 and Beyond

INITIAL TRUST ASSESSMENT

Is The Existing System Supportive of Trust Requirements?

- Hardware Base
- Software Architecture
  - Structure
  - Access Mediation
- Documentation

- **How to Mediate Access and Enforce MAC?**
  - **What Comprises TCB?**
  - **What Are Subjects?**
  - **What Are Objects?**
  - **How Is Labeling Done?**
  - **How Can Data Flow Between Subjects and Objects?**
  - **How Is Mediation Done?**

- **Are the TCB Structure and Characteristics Adequate?**

  Does the TCB:

  - **Have Domains for Its Own Execution?**
  - **Maintain Process Isolation Through Distinct Address Spaces?**
  - **Exclude Non-protection Critical Elements?**
  - **Support Least Privilege?**
  - **Have A Layered and Modular Structure?**
  - **Embody A Conceptually Simple and Complete Protection Mechanism?**

- **Are the Remaining Trust Features Supported?**

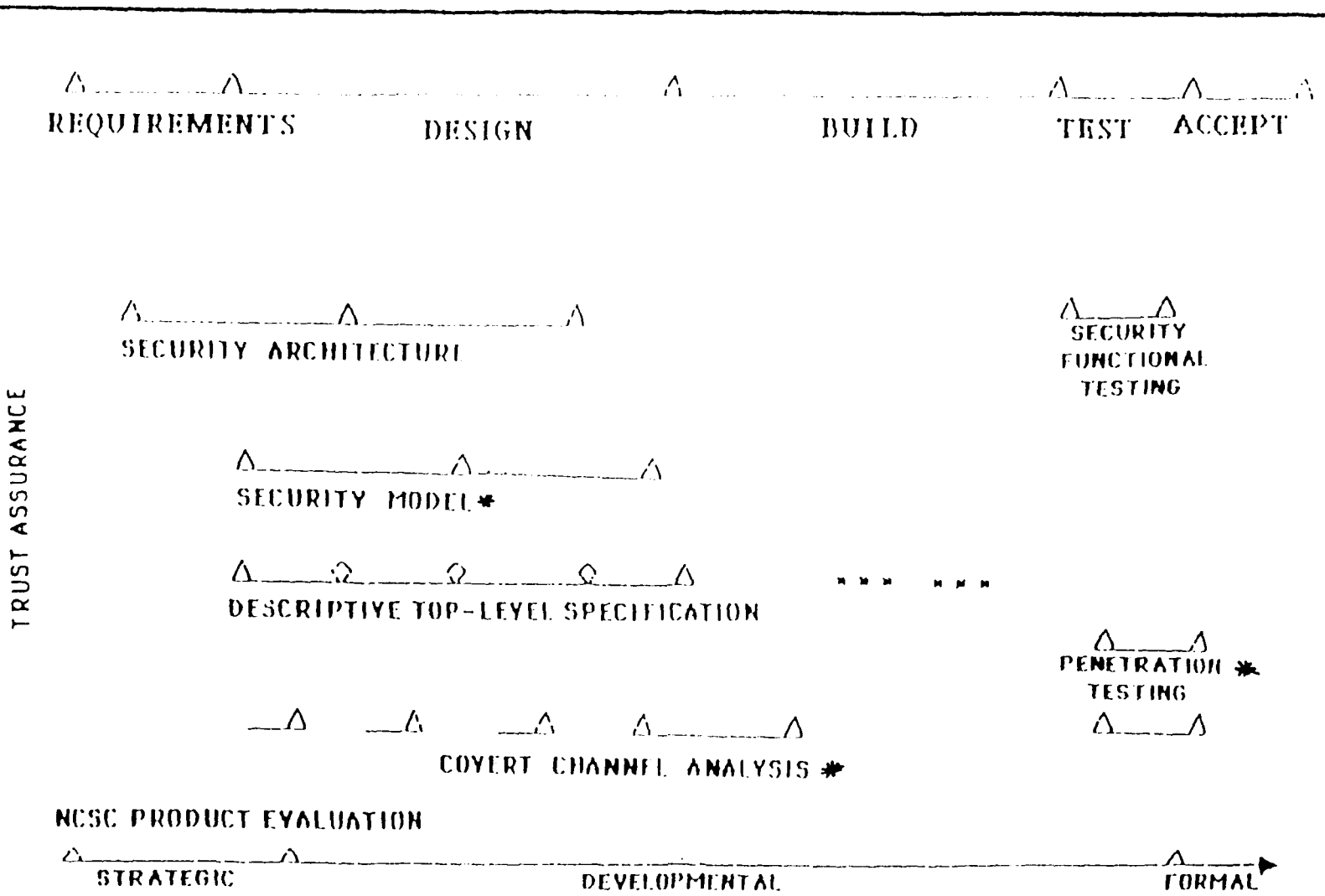
  Does the System Have Adequate Mechanisms for:

  - **Identification and Authentication?**
  - **Trusted Path?**
  - **Audit?**
  - **Security Administration Function?**
  - **Object Reuse?**
  - **Trusted Recovery?**

What Are the Covert Channels?

TRUST ENGINEERING

- Provides Focus on Trust Throughout Development
- Integrates Trust-Related Activities with Development
- Manifests Tension Between Trust and Other Requirements
- Requires Developers' Cooperation and Endorsement



TRUST ENGINEERING ACTIVITIES

TMACH

- Trust Assessment
  - Amenable System Structure
  - Strong Access Control Mechanism
- Development of System Protection Strategy
- Policy Representation for Kernel
  - Modeling
  - Covert Channel Analysis
- Build for Kernel
- Policies for Servers
  - Trust Assessment of Design
  - Modeling
  - Covert Channel Analysis
- Build for Servers
- System Testing
  - Security Functional
  - Covert Channel
  - Gedanken Penetration

Incorporating Trust into an Existing System

Spiral Process Model Version

1st Loop: Is System Amenable to Trust?

- At Reasonable Cost
- With Some Compatibility

Risk Resolution: Perform Trust Analysis to Size Effort

- Locate and Clarify Trust Issues
- Determine Modification Costs and Impact

Decision: Target Level

Whether to Proceed

2nd Loop: Develop Protection Strategy

- Meet Target Level
- Compatibility
- Cost
- Performance

Risk Resolution: Develop Design with Mandatory Access Mediation

- Trust Analysis
- Modeling
- Covert Channel Analysis

Decision: Can Design Comply with Constraints?

- Continue as Planned
- Change Goals
- Kill Project

3rd Loop: Design and Develop Trusted System

"Traditional" Development

Prototype

Waterfall

.
.
.

TRW

Attachment #5

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Agenda

ASOS Questions and Answers ..... Lou Nagy

Computer Security Lessons Learned ..... Pat Rougeau

Risk Management for a Trusted Application ..... Dan Sterne

Trust Engineering for Commercial Products .... Marty Branstad

Lunch

Process Model Brainstorming ..... All

- Review notes from 2/6/89 meeting and refine
  - Drivers
  - Constraints
  - Other trade-offs
  - Trust dimensions
- Distinctions between commercial and mission-critical trusted systems (initial analysis only -- more next meeting)
- Define cycles of spiral for trusted systems
  - Risks to be identified/managed
  - Activities
  - Documents
  - Criteria for next phase transition

(Deferred to next meeting)

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Participants:

Ann Marmor-Squires, TRW
Pat Rougeau, TRW
Bonnie Danner, TRW
Lou Nagy, TRW
John McHugh, CLI
Marty Branstad, TIS
Dan Sterne, TIS

Activities

- Review and modification of notes from 2/6/89 meeting
- Refinements to process model
  - Drivers (Ada, trust, performance)
  - Trust dimensions
  - Constraints (political/sociological, cost, available technology/knowledge)
  - Other tradeoffs against trust

General process model definition:

- Conceptual framework for a system lifecycle which structures the activities associated with the creation and support of the system

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

General Process Model Goals

- Useful for enhancements to existing systems as well as new systems (i.e., multiple entry points to the process)
- Supports "cradle to grave" development
- Promote product quality
  - Reliability
  - Mission-effectiveness (including performance)
  - Trustworthiness
  - Maintainability
- Be cost-effective
- Be practical
  - Easy to use
  - Training available
  - Comprehensible results
  - Tailorable to customer constraints
- Accommodate diverse set of lower level development paradigms (e.g., prototyping, reusable components, program families, evolutionary development)
- Provide visibility into the process (managers, customers)
  - Progress being made
  - Likelihood of success
  - Measurable
- Aid predictability and control
  - Awareness of status
  - Achieving convergence
  - Transitioning to next stage
  - Supported by configuration management
- Ability to respond to changing requirements
  - Risk-driven
- Support traceability throughout the process
- Amenable to automation as appropriate
- Scales up to complex systems

[Underlined are new]

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

General Process Model Components

- Guidance and alternatives to be considered
- Policy and standards to be followed
- Well-defined activities (multiple threads possible)
  - Methods
  - Procedures
- Completion criteria: transitioning to next phase
  - Reviews
  - Metrics
- Constraints
- People
  - Project
  - Management
  - Customers
- Tools to effectively manage and support the process
- Documents

[merger of two lists]

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Process Model for Trusted Systems in Ada

- Basis
- Drivers
- Constraints

Basis: Spiral Process Model

- Risk management (resolution of drivers against constraints)
  - Risk management plan
  - Perform risk assessment for each activity
  - Focus on specific risks -- identify risk(s) and then mitigate it through specific activities, e.g., prototyping, specifying, simulating, formally modeling/analyzing
- Iteration at different levels
  - Controlled and managed through automated configuration management

Drivers

- Trust
- Ada
- Performance

Constraints

- Political/sociological environment
- Cost
- Available technology/knowledge

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Trust as a Driver

Continuum of trust levels -- discussion on what level(s) we'll address in this contract

1. TCSEC definition of trust
   - Confidentiality
   - Address "beyond B2 criteria"

2. Security - * =
   - Confidentiality
   - Integrity
   - Assured service

-
-
-

n. Functional correctness

Discussion

- Defining trust
- Defining trust policy
- Coupling policy with assurance
- Coupling mission criticality with trust
- Determining impact of trust
- Impact of complexity on trust
- Controlling iterative development in a trusted manner

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Trust as a Driver

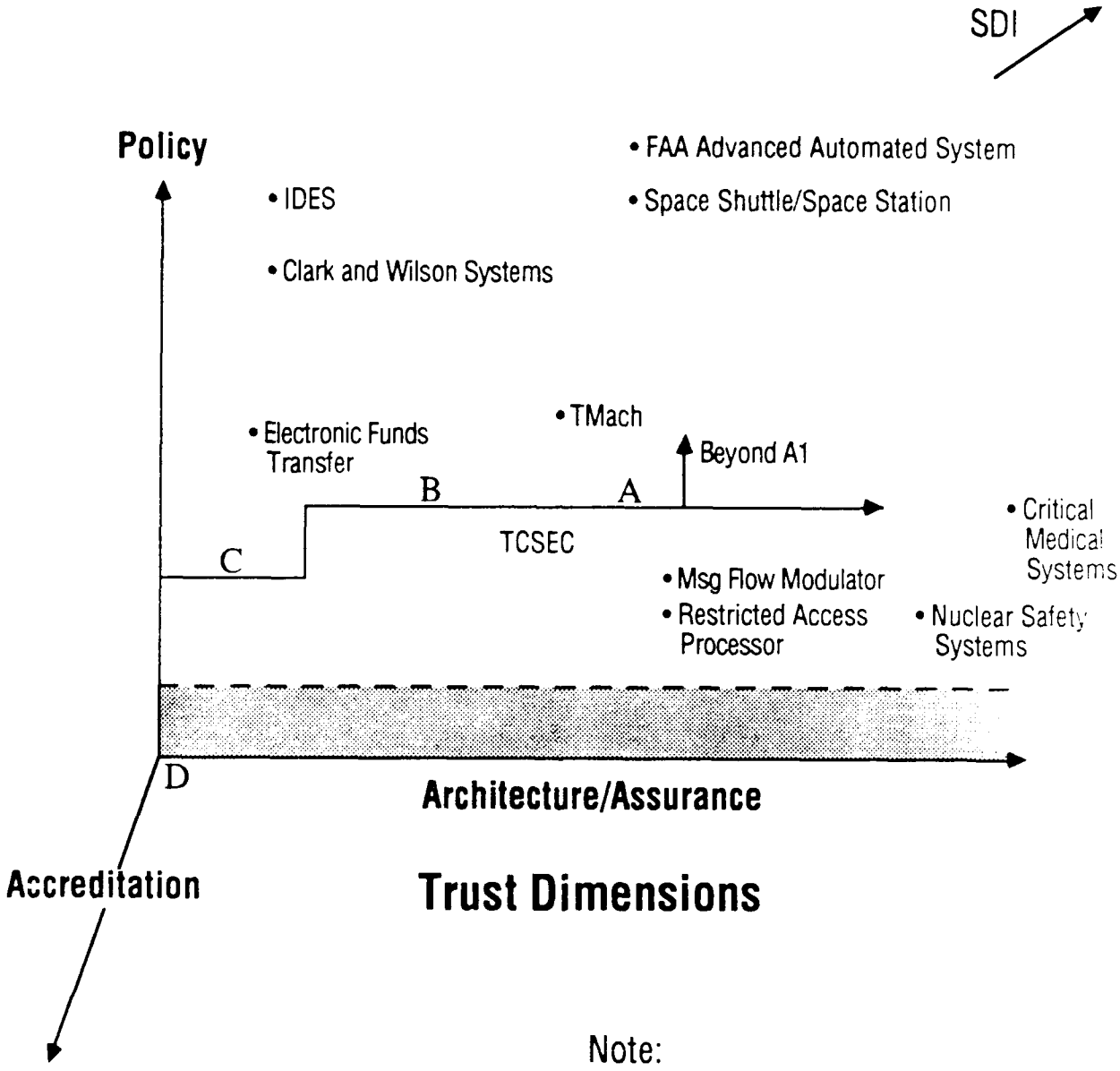
Aspects of Trust

- Security (confidentiality)
- Safety
- Mission-criticality
- Data integrity
- Process integrity
- Assured service
- Functional correctness
- Reliability
- Evolution of trust

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989



Trust Dimensions

Note:

Metric on policy not defined:

- Complexity?
- Encompassing more aspects of trust?
- Degree of formality?

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Ada as a Driver

- Uniformity offered by Ada in specification/design and implementation
  - Supports traceability (no gap between notations)
  - Provides consistent metric throughout life cycle
  - Early design validation (compilable Ada specifications) and better interface definition/ checking
- Encourages software engineering aspects of Ada
  - Top down development
  - Object oriented design/development
- Reusability of Ada packages, components, designs
  - Promotes cost-effectiveness
- Increasing opportunity to use tools:
  - Support reasoning needed for trust
  - Support configuration management control needed for trust
- Verifiability of Ada:
  - Richest subset for target systems we'll address

Issue:

- Use of Ada as a formal specification language to support consistent notation throughout the development process

[Get information on Dave Luckham's work and on Odyssey Research Associates work w.r.t. formal specifications with Ada-like languages]

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Performance as a Driver

Aspects of performance

- Speed
- Effectiveness
- System performance
- Mission-criticality performance
- Resource ulilization
- Real-time effects

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Constraints

Political/Sociological Environment ("Human Aspects")

- Communication methods
  - Within project
  - Up to management
  - With customers
  - With subcontractors
  - With evaluators
  - With accreditors
  - Between security/trust engineering and developers
- Organizational constructs
  - Company (standards and procedures, culture)
  - Customer environment
- Evaluation/certification process (NCSC, other)
- Accreditation/acceptance criteria
- Political ramifications
  - What's acceptable to the organization
  - What's acceptable to the evaluation/accreditation
  - Issues and differing goals among customers, evaluators and accreditors
- Staffing continuity and stability
  - Project (different for commercial products and mission-critical systems developments)
  - Customer community
  - Evaluation/certification/accreditation communities
- Physical environment
  - Location of people
  - SCIFs
  - Optimal equipment mix
- Response to change in perception of requirements (e.g., Ada is now important. . . , new aspect of trust is now important . . ., etc.)
- User community
  - Usually different from the customers
- Skill mix on projects
  - Ada expertise and experience
  - Trust expertise and experience
  - Much skill specialization is reality (e.g., developer, security engineer) but cross-fertilization is what's needed
- Methods of conflict resolution
  - Strength of personality

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Constraints

Cost

- Process needs to be cost-effective
- Financial rules
  - Capital vs. labor dollars
  - My $ vs. your $
- More cost up front -- lower lifecycle cost
- Mesh cost model with revolutions of the spiral
- Appropriate use of assurance technology consistent with cost
- Schedule

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Constraints

Available Technology/Knowledge

[Some aspects not well understood]

- Available products/components
- Available support tools
  - Verification environment
  - Development environment
- Maturity level
  - Technology
  - Knowledge
- Trustworthiness of the development environment
  - Compiler
  - Ada programming support environment (APSE)
  - Configuration management system

DARPA
Advanced Computing Systems Project

Notes from Process Model Brainstorming Session

17 February 1989

Issues to be Addressed

- Domain of target systems -- types of systems we are addressing
  - Large complex mission-critical systems?
  - Systems like FAA's new Air Traffic Control System?
  - Trusted commercial products?
- DoD-STD-2167A
  - Consistent with? probably not
  - Evolvable from it to our process model?
- Trust issues with respect to the development environment
- Other system tradeoffs against trust
  - Evolvability
  - Extensibility
  - Configurability
  - Applicability
  - Complexity

TRW

Attachment #6

DARPA
ADVANCED COMPUTING SYSTEMS (ACS) PROJECT

Revised Notes on
Distinctions between Commercial
and Mission-critical Trusted Systems
13, March 1989

An ACS Process Model goal is to provide a flexible framework that applies to both commercial and mission-critical trusted system life cycle activities. To better understand the process model applicability to both environments, distinctions between them were analyzed. Figure-1 presents an overview of the distinctions. A summary of the initial analysis is provided below.

BASIC DRIVERS

Basic drivers for commercial and mission-critical projects for which the process model applies are similar: Ada, trust and performance. However, the motivations are different.

Commercial Systems are motivated by commercial markets and upward compatibility with corporate product lines. Market pressure is a major influence.

Mission-critical Systems are motivated by mission requirements and are normally focused on new system development. While major upgrades are sometimes developed and compatibility issues are of concern, total replacement is the usual situation. Government requirements are beginning to change, however; and there will likely be more emphasis in the future on upward compatibility. Mission need is a major influence.

Ada. In response to Government mandates to use Ada in DOD procurements, both commercial and mission-critical system developers are increasing their Ada-based capabilities.

> **Commercial Systems** are responding to the market needs of the Government and contractors. Commercial Ada products are evolving. However, there is no current large-scale commitment to Ada products.
>
> **Mission-critical Systems** are more frequently driven by the Government mandate for Ada. Many current and future procurements have the Ada language as a hard requirement for mission-critical development. As the Government commitment to Ada grows, the importance of Ada capabilities is increasing throughout the contractor community. Contractor involvement in Ada systems development is already significant, and it is growing.

	Commercial Systems	Mission-Critical Systems
Drivers:		
Ada:	• Government emphasis • Government and contractor markets • Limited involvement	• Government requirement • Increasingly important • Broad-based involvement
Trust:	• Market pressure • Need for certified products • Current emphasis on security	• Critical Government requirements • Accreditation emphasis • High risks imposed
Performance:	• Product upgrades • Competitive edge • Flexible requirements • Product upward compatibility	• Contractual obligation • Mission needs • Little or no requirements flexibility • New or replacement system
Constraints		
Cost:	• Corporate marketing basis • Company managed and controlled • Funding based on business perceptions • Company-driven expectations • More staff/management stability • Fewer levels of diversity • More realistic cost planning • Government-driven certifications • Cost emphasis on coding and testing • Responsive to market changes	• Government-based contract • Subject to Government management and control • Based on Government funding • Mission oriented • Constrained by politics, federal budget • Complex organizational structure • Less stability at all levels • Cost goals can be unrealistic • Government-driven accreditations and certifications • Cost emphasis on coding and testing tied to Government reviews • Responsive to perceived mission needs
Political/Sociological Environment:	• Simpler management practices • Company-based politics • Flexible personnel practices • Continuity of corporate knowledge • Centralized technical information • Communication easier • Requirements can be responsive • Commercially-based goals • Enhancements emphasis • Certification involves the government	• Complex management structure • Government and company politics • Staffing tied to SOW and costing • High turnover, less continuity • Distributed knowledge base • Communication more difficult • Requirements tied to Government perceptions • Mission-based goals • New procurement emphasis • Need to agree on accredition and certification requirements
Available Technology/ Knowledge:	• More focused technology base • Unified project support tools and methods • Participating in standards developments and new initiatives • Not likely to push state-of-the-art • Reality based • Not generally high risk	• Diverse technologies • Project-oriented support tools and methods • Responding to existing standards • Responding to Government desire for advanced technology • Often pushing the state-of-the-art • Government requirements may not be realistic • Can often be high risk

Figure - 1. Distinctions between Commercial and Mission-Critical Trusted Systems

Trust. Trust and security needs are creating more business opportunities for both commercial and mission-critical system developers. Trust and security issues are evolving as a result of enhanced computer and communications capabilities as well as the growing need for more complex software systems.

Commercial Systems developers are experiencing increased market pressure to meet Government and industry requirements for trusted systems. There is a genuine need for certified products, and there is a defined customer base. Evaluated products are emerging in response to Government DOD requirements. Other trusted products are being developed for the trusted systems market.

Mission-critical Systems are frequently required to meet trust requirements defined by the Government. Developers of systems undergoing certifications and accreditation are strongly driven to provide acceptable levels of trust with enough assurance to convince the responsible authorities that the system meets the trust requirements. Government requirements are not always based on the realities of today's technology limitations. Consequently, trust requirements may represent high risk areas for a system.

Performance. Performance has been a traditional driver for system developments both in the commercial and mission-critical worlds. Commercial systems are driven by the need to be competitive, and they often upgrade existing products for performance enhancements alone. Mission-critical systems are developed to meet strict Government requirements for system performance.

Commercial Systems base performance requirements on commercial considerations. Some product upgrades and new products are built to meet market demands for better performance. There can be great flexibility in performance requirements since they are founded on perceived marketability. Upward compatibility of existing products is an important issue for commercial developers.

Mission-critical Systems must respond to critical performance requirements established by the Government for many procurements. Mission operational needs are paramount with little or no flexibility permitted for some systems. A contractual obligation to meet Government requirements drives the system design and development. Realistic trade-offs between security and performance are frequently necessary in trusted system developments. Performance and security risks must be identified early. Most mission-critical systems are new procurements in response to new or increased mission requirements.

CONSTRAINTS

Constraints vary greatly for commercial and mission-critical trusted

system projects. Major differences in cost, available technology and sociological considerations are summarized below.

Cost. Cost is a critical factor that influences both commercial product development and mission-critical systems.

Commercial Systems are constrained by corporate marketing strategies, corporate management, company willingness to finance projects and company strength in the market place. Cost reporting and controls are likely to be much looser than Government funded projects. Projects are subject to internal policies and audits with no need to comply with Government project management costing structure, audits and accounting practices. Cost planning is usually realistic and tied to feasible goals and market perceptions. Projects can be suddenly terminated as a result of corporate business needs. Additional funding can be obtained, also in response to business needs. Unlike mission-critical projects, there is a flexibility for spending money on commercial product development with repeated product sales as a pay off for the initial investment.

Reporting and documentation cost constraints are limited to corporate and market needs. Traditionally, design documentation is minimal while a heavy emphasis has been placed on user documentation. Assurance costs have largely been applied to testing. With the move toward developing (TCSEC) trusted products which require Government (NCSC) product evaluation, commercial projects must respond to the design documentation and assurance requirements of the Government.

The commercial model for project cost distribution is likely to vary for different vendors. There is an emphasis on the coding and testing costs rather than design. Some companies that are emphasizing computer aided systems engineering (CASE) methods and tools may become notable exceptions.

Commercial projects are cost-driven by competitive factors and the need to bring a product to market.

Mission-critical Systems are constrained by the Government defense budget, congressional allocations and perceived defense needs (politics). Costing is tightly controlled by Government requirements for cost management, and projects are required to provide detailed reviews and reports to the Government. All projects are subject to basic Government controls and audits while the specific costing structure depends on the type of contract in place. Mission-critical projects are subject to sudden reductions or terminations based on Government budget needs (e.g., Gramm-Rudman, new administration, etc.)

Reporting and documentation consume a large portion of project budgets due to traditionally heavy Government requirements for

mission-critical systems that must be operated and maintained by the Government. Design documentation is emphasized, although traditional assurance measures still rely largely on testing activities. Trusted system development, however, requires a sizeable investment in design and assurance activities for both certifications and system accreditation.

The current cost model for mission-critical system development is tied to traditional Government practices. The distribution is as follows: 20%-design, 40%-code, 40%-integration and test. A process model for trusted systems requires an emphasis on risk control and a heavier investment up front. A recommended distribution is: 50%-design, 25%-code, 25% test and integration.

Mission-critical systems are driven by Government cost constraints, project control dictates and mission requirements.

Political/Sociological Environment. The human aspects of project activities are similar in many respects; however, commercial and mission-critical projects are very different in orientation, politics and personal dynamics. Issues such as staffing, physical environment, communication, perception of requirements, customer satisfaction and methods of conflict resolution are not viewed and managed the same way in commercial and mission-critical projects.

Commercial Systems generally have a simpler management and control atmosphere with politics that involve company roles and corporate expectations. The management-staff relationship is based on company culture and individual dynamics. Commercial personnel practices are tied to company-driven perceptions of value and tend to be very flexible. Most commercial projects enjoy staff stability and a continuity of corporate knowledge.

Communication is a challenge for any complex system involving many personnel working on different aspects of a project that must ultimately form an integrated whole. Within a commercial atmosphere there are generally fewer levels of diversity (e.g., no Government or IV&V oversight) and communication should be easier to achieve.

Commercial systems are more likely to respond to established product lines. Requirements are defined to satisfy a perceived market as well as to identify a technical set of needs for a specific product. There is generally more flexibility for the development of system features. Commercial requirements are more reality-based and tend to respond to known and proven ideas. Government requirements are an important consideration for trusted products. Certification/evaluation planning involves Government oversight and assistance with product definition. Products are built for flexibility, and a usual goal is certification for use in various environments. The complexities and concerns of mission-operation requirements and system

accreditation are not a part of most commercial product projects. Commercial systems must be responsive to market changes above all.

Mission-critical Systems have a complex management structure with layers of responsibility within the Government and the company (or companies) involved in a project. Reporting and project controls are usually formal and based on strictly defined Government requirements as well as corporate project management rules. Personnel practices are tied to Government needs and perceptions since labor categories and task definitions are driven by the project statement of work and cost management policies. Therefore, there is less flexibility for staff and project management than in a commercial project environment. Due to the complex political environment, the strict controls and the nature of the Government (especially the services), staff turnover can be a frequent occurrence on both the Government and the contractor side.

Government oversight may involve diverse organizations, each playing some role in the system acquisition, accreditation, operations or maintenance. Communication issues and political considerations can be extremely difficult to resolve. In addition, high staff turnover can complicate communication for a mission-critical project.

Requirements must be satisfied in response to Government perceptions; therefore an understanding and agreement must be achieved and maintained between the Government and the responsible contractor. Government requirements for a trusted project involve trust certification and system accreditation. Security requirements must be resolved to the satisfaction of all responsible authorities. Accreditation issues need to be well understood and defined at the beginning of the project. Mission-critical systems are usually built for a specific environment and oriented toward the operational mission. Above all, mission-critical systems are responsive to customer mission requirements and customer perceptions.

Available Technology/Knowledge. While commercial and mission-critical trusted system projects are both limited by security and technology issues that are still in the research stages (e.g., formal assurance methods beyond confidentiality and at code level and for large systems), there are differences in the way current technological advances are approached and accessed.

Commercial Systems developers are likely to have access to focused corporate knowledge and a unified set of project support tools and methodologies with available guidance from corporate personnel. For example, standard, commercial configuration management should be stronger than mission-critical system configuration management which probably differs for each

project. In general, a commercial environment provides a production-oriented basis for technical advancements specific to the corporate activities, strategies and products derived for perceived markets. Products requiring high risk advancements are usually not undertaken. Decisions are based on expected return weighted against risks. Technological advances are sought to provide a competitive edge.

Many companies building commercial products are participating in standards definitions (e.g., POSIX, SDNS, GOSIP) in cooperation with the Government and other vendors. Government and industry pressures for standards have forced many vendors to be actively involved, while developers of mission-critical systems have not been as oriented toward standards definitions.

Mission-critical Systems developers have access to available corporate technology. However, much of the available knowledge, methods and tools are widely distributed and project oriented. Much corporate knowledge is based on one-of-a-kind applications that are founded in project-specific details. Many tools are used only for a specific project and are not readily tailorable. The corporate technology base may be advanced and very strong in a specific area while lacking in broad applications. In general, mission-critical systems are built to satisfy operational needs and are responding to standard requirements rather than establishing and driving the development of standards. Systems with high trust requirements may be driven by unrealistic Government demands that stretch the state of the art. Technological advances can be mission-driven and accomplished in response to the strict demands of the Government.

TRW

Attachment # 7

ADA/SECURITY STUDY AND MACH ANALYSIS

Internal Working Note
13 March 1989

INTRODUCTION

This internal working note is based on initial work carried out by L. Nagy of TRW over a seven week period (23 January to 10 March 1989) before we received our SUN hardware and MACH software. It is a snapshot of activity related to defining Appendix B of the Phase I Process Model Report ("Ada/Security Study and MACH Analysis") and in gaining understanding of the current MACH and TMACH work at CMU and TIS. Some of the analysis is based upon discussion with the TIS TMACH developers and analyzing their TMACH development at TIS facilities in Glenwood, Maryland.

OVERVIEW

A. Task Definition

In the Technical Volume of our Proposal prepared 2 July 1987, this task is defined as follows:

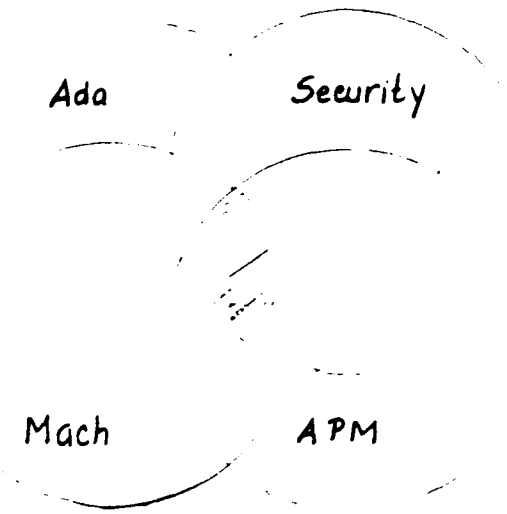
> "Analysis work will be directed at providing a formal Ada/Security analysis for MACH. This analysis will include development of external interface specifications in Ada for MACH. The specification will be used to identify the Ada and security issues that must be resolved in development of a trusted MACH prototype in Ada. This analysis and specification will rely on the Process Model developed with this effort and will provide an opportunity to identify the practical issues that require improvements."

Initial trust assessment of MACH has been carried out by TIS under a separate DARPA contract as part of their development of a proof of concept TMACH prototype in C. Our work will build upon TIS's work and identify and evaluate issues related to the development of trusted MACH in Ada. We also will extend our analysis to the development of tangible products (Ada specifications, prototypes, tools) which can be directly applied toward the ultimate goal of trusted MACH in Ada.

The areas of interest are in the interaction of the following domains:

- Ada
- Security
- MACH (and trusted MACH)
- Advanced Process Model (APM)

and illustrated in the following figure:



Domains of Interest

If we consider the areas of interest two at a time, we may note that in some areas of overlap, i.e., Ada-Security, there is already a relatively long history of interest and some effort (although significantly less of achievement). Other areas of overlap (APM-Ada) and (APM-Security) are being addressed by other sections of our report. Still other areas are being covered, at least partially, by related efforts: TIS in the area of MACH and Security, and CLI in the area of Ada and security.

The one area of overlap that as yet hasn't gotten significant attention is the Ada-MACH one and as such this will be the primary focus for this part of the effort. It should be noted that this area of overlap includes portions of other areas of overlap and as such the results and issues from these areas are also applicable.

B. Activities

The effort is being structured by first examining the MACH-Ada overlap. This area roughly corresponds to the language issues of Ada as applied to MACH development. The security domain will then be factored in, in essence, examining the security issues related to developing MACH in Ada. Finally, the Advanced Process Model domain will be factored in. This effectively introduces software development lifecycle factors to the primary objective.

Note that the order of introduction of the domains is important. The order is intended to reflect the relative importance of these factors in this effort's resolution of the issues of the primary objective.

1. MACH and Trusted MACH Analysis

The objective of this task has been to gain a deeper understanding of MACH and Trusted MACH. To be included in this understanding is the overall functional capabilities of MACH and TMACH, the design of key subsystems of MACH and TMACH, and the implementation details of MACH and TMACH. The purpose of this understanding has been to gain insight into the project's ultimate objective of examining and evaluating the issues related to the design and development of a trusted MACH in Ada.

The initial analysis is described here:

- It appears that the current MACH implementation is a highly intertwined composite of MACH and Unix elements. Included in these elements are relatively few TIS modifications as part of their trust enhancement of MACH. Not all MACH capabilities are present (e.g., threads). All Unix features appear to be present although reliability of the underlying systems may not be as good as the original Unix system.

- The MACH and TMACH implementations make strong use of MACH IPC capabilities and any Ada implementation of MACH will have to be efficiently done.

- The MACH kernel is currently implemented using Unix kernel processes. Any trusted implementation using this design will have to be able to formally handle this abstraction.

- The current MACH development environment is highly Unix and C based. Additional CMU tools are also utilized in the maintenance and generation of the MACH kernel and server executables. This development environment is driven by portability concerns target dependencies, and existing Unix code. Any Ada environment for MACH development should likewise provide features for:

    a) Isolating implementation which is non-portable
    b) Selecting particular targets

- The MACH Memory Management subsystem is being considered as a possible development task in a later phase (along with previously mentioned candidates such as the Name Server). The size of the Memory Management subsystem and the TIS Name Server are similar. Moreover, an Ada implementation of this subsystem would allow for the addressing of high-risk issues of performance impact, portability, and security issues of this critical kernel component.

- Because of the complex detail required to construct an RPC based interface between servers, the MACH environment provides a tool, MIG (MACH Interface Generator), which automatically generates the code for receiving and sending messages. MIG generates C RPC procedures for sending and receiving MACH messages based on a PASCAL-like specification of the message.

  An extension of this approach for generating Ada interfaces (and code) from these specifications is being considered.

2 Ada and Ada/Security Analysis

The objective of this set of activities is to examine and evaluate those features of Ada affecting the portability, performance, cost, compatibility and other aspects of an implementation of MACH in Ada. An identification and analysis will be made of those features of Ada whose use in trusted systems has raised concern in the past.

The results obtained thus far are:

- The allowed syntax of MACH messages (essentially arbitrary structures) will pose problems in an Ada implementation of the MACH kernel. This is a direct result of the necessity of the MACH kernel to be able to parse each and every message sent (to mediate port transfer) and Ada's inability to effectively manipulate the underlying representation of an object.

- Many complaints have been voiced about the impreciseness of some of Ada's language features. These features will be enumerated and annotated with the impact on security/ verification of Ada programs.

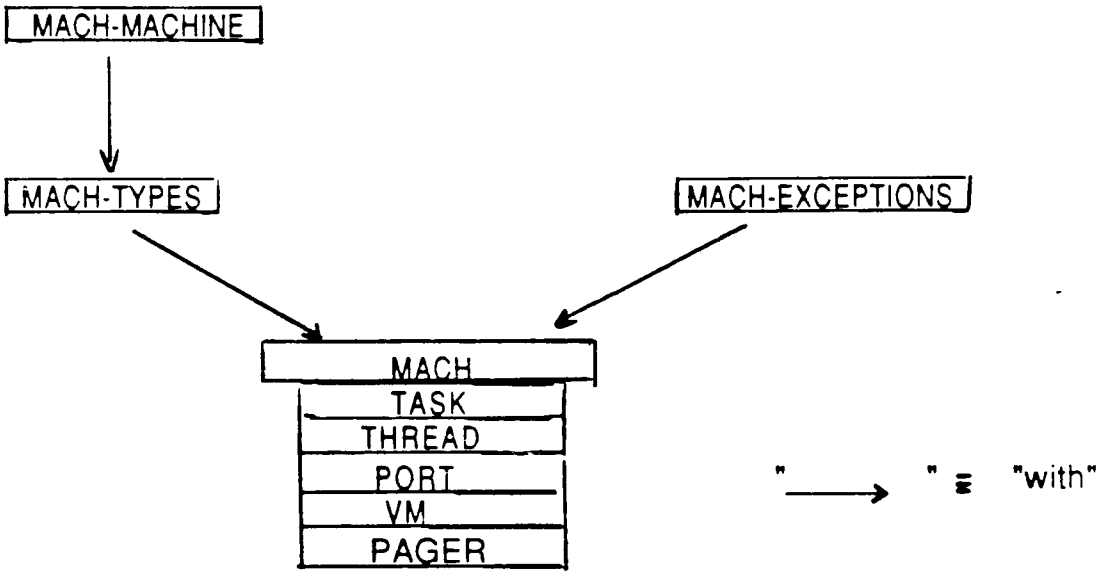
- An initial specification of an Ada interface specification and body to the MACH kernel is being done. The goal is to produce a compilable and executable Ada interface to the MACH kernel. This implementation will be evaluated with respect to security, performance, and portability. The structure of this interface is:

MACH Kernel Interface in Ada

The following packages will be defined:

MACH-MACHINE:	defines all hardware/system/configuration dependencies
MACH-TYPES:	defines all basic kernel abstractions which one global to the five subsystems
MACH-EXCEPTIONS:	defines all the kernel exceptions returned by the MACH calls
MACH:	defines 5 nested packages: TASK, THREAD, PORT, VM, and PAGER. These packages provide the interface to the kernel provided services according to subsystem

These packages will have the following dependency structure:



3. PROCESS MODELS

The objective of this set of activities has been to examine the constituent activities and products of a software development life-cycle with particular emphasis on Ada and trusted system development.

Process Model examples being investigated are: the Ada Process Model, the Waterfall Model, Formal Development paths, Orange Book requirements on trusted system development, and development environments (TRW's Quantum Leap and the Arcadia Project).

The initial analysis indicates:

- The package specification concept is the key Ada feature used in developing Ada systems and has a use which transcends the implementation language.

- A significant problem in the past with the development of trusted systems has been the establishment and maintenance of a correspondence between the formal and functional specifications. This appears to be the result of the amount of change that a functional specification undergoes during development and the fact that the formal and functional specifications are expressed separately.

C. ACTIVITY PLANS

Based on the results presented in the previous section and the overall objective of developing a trusted version of MACH in Ada the following will be included in the next phase of activities:

1. MACH and TMACH Analysis

   - Continue with the MACH kernel evaluation

   - Continue with the evaluation of enhancing MIG to support the generation of Ada code or interfaces

2. Ada and Ada/Security Analysis

   - Complete an Ada package specification and rationale for the MACH/TMACH kernel interface.

   - Complete an annotated list of Ada features which affect security of an implementation. Include possible approaches for dealing with these insecure features.

3. Security

   - Continue to examine the current and research practice of trust engineering with the goal of identifying technology (primary tools) which may be applicable to trusted MACH development in Ada.

   - Based on this list, attempt to identify areas of critical need.

4. Process Models

   - Investigate the use of a common specification language related to Ada.